TABLE I. RUN TIMES OF THE CONSIDERED IMPLEMENTATIONS

| Algorithm | Query | | |
|---|---|---|---|
| | Star | Triangle | Path |
| Naive | 84 ms | 1689 ms | 5235 ms |
| Baseline | 73 ms | 685 ms | 4039 ms |
| FastGFDs | 22 ms | 470 ms | 1321 ms |

TABLE II. MEMORY CONSUMPTION OF THE CONSIDERED IMPLEMENTATIONS

| Algorithm | Query | | |
|---|---|---|---|
| | Star | Triangle | Path |
| Naive | 35956 kB | 35952 kB | 35952 kB |
| Baseline | 172656 kB | 172656 kB | 172652 kB |
| FastGFDs | 35956 kB | 35956 kB | 35952 kB |

these algorithms run out of memory, disk swapping might occur, leading to a drastic decline in performance. In this regard, the original algorithm is in an inferior position, as it reaches the memory limit more quickly, given that it requires five times more memory than the others. Additionally, in our experiments, we used a small graph, resulting in a negligible CPI memory footprint. However, increasing graph size could change this outcome and potentially turn it into a serious limitation. To address this issue, further experiments with larger graphs are required. We also plan to explore this in our future work.

- Next, the performance of algorithms heavily depends on the query and dataset properties. In order to negate this threat, we have tried to evaluate these algorithms using several different queries.
- Finally, in our implementation, we relied on the Boost graph representation. Switching to another library for representation, or implementing a custom one, could affect the result. However, we believe that it might affect only the obtained ratios, but not the relative order of implementations. Exploration of this issue is also a matter of future work.

## VII. CONCLUSION AND FUTURE WORK

In this paper, we described our open-source implementation of the graph functional dependency validation algorithm from study [1] and an improved algorithm with fast graph pattern matching that targets specifically a low-end multi-threaded environment. Performance evaluation run on a real-life graph demonstrated that the improved algorithm shows up to three times performance (2.6x on average) increase over the state-of-the-art algorithms. Employing the new subgraph matching algorithm also led to a fivefold reduction in memory consumption.

The source code of our C++ implementation and evaluation datasets can be found in the GitHub repository (https://github.com/Mstrutov/Desbordante/pull/154) of the Desbordante project [10].

There are several directions for future work:

- Integrate the CPI-based subgraph matching into the parallel scheme of the original algorithm. However, the parallel scheme is designed to work with small graphs, while CPI is more suited for large graphs. Therefore, there should be a some kind of compromise on the graph size.
- Develop a special parallel scheme for the CPI-based subgraph matching. Since the previous approach might not yield positive results, a straightforward idea is to parallelize the CPI-based querying.

## ACKNOWLEDGMENTS

We would like to thank Anna Smirnova for her help with the preparation of this paper.

## REFERENCES

[1] W. Fan, Y. Wu, and J. Xu, "Functional dependencies for graphs," in *Proceedings of the 2016 international conference on management of data*, 2016, pp. 1843–1857.

[2] T. Papenbrock and F. Naumann, "A hybrid approach to functional dependency discovery," in *Proceedings of the 2016 International Conference on Management of Data*, ser. SIGMOD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 821–833. [Online]. Available: https://doi.org/10.1145/2882903.2915203

[3] W. Fan, C. Hu, X. Liu, and P. Lu, "Discovering graph functional dependencies," *ACM Trans. Database Syst.*, vol. 45, no. 3, sep 2020. [Online]. Available: https://doi.org/10.1145/3397198

[4] F. Mahdisoltani, J. Biega, and F. Suchanek, "Yago3: A knowledge base from multilingual wikipedias," in *7th biennial conference on innovative data systems research*. CIDR Conference, 2014.

[5] F. Bi, L. Chang, X. Lin, L. Qin, and W. Zhang, "Efficient subgraph matching by postponing cartesian products," in *Proceedings of the 2016 International Conference on Management of Data*, 2016, pp. 1199–1214.

[6] J. Yang and W. Jin, "Br-index: An indexing structure for subgraph matching in very large dynamic graphs," in *Scientific and Statistical Database Management: 23rd International Conference, SSDBM 2011, Portland, OR, USA, July 20-22, 2011. Proceedings 23*. Springer, 2011, pp. 322–331.

[7] G. Chernishev, M. Polyntsov, A. Chizhov, K. Stupakov, I. Shchuckin, A. Smirnov, M. Strutovsky, A. Shlyonskikh, M. Firsov, S. Manannikov, N. Bobrov, D. Goncharov, I. Barutkin, V. Shalnev, K. Muraviev, A. Rakhmukova, D. Shcheka, A. Chernikov, D. Mandelshtam, M. Vyrodov, A. Saliou, E. Gaisin, and K. Smirnov, "Desbordante: from benchmarking suite to high-performance science-intensive data profiler (preprint)," 2023. [Online]. Available: https://arxiv.org/abs/2301.05965

[8] J. G. Siek, L.-Q. Lee, and A. Lumsdaine, *The Boost Graph Library: User Guide and Reference Manual, The*. Pearson Education, 2001.

[9] B. Rozemberczki and R. Sarkar, "Twitch gamers: a dataset for evaluating proximity preserving and structural role-based node embeddings," *CoRR*, vol. abs/2101.03091, 2021. [Online]. Available: https://arxiv.org/abs/2101.03091

[10] A. Chernikov, "Desbordante gfd pull request," https://github.com/Mstrutov/Desbordante/pull/154, 2023.